IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Oct 2022

Ethereum Smart Contracts

Dr. Srividhya Murali and Dr. Sapna Sharma
S. K. College of Sci & Comm., Nerul, Navi Mumbai
vidmur@yahoo.com

ABSTRACT

Smart contracts are computer protocol that can digitally check, apply, and enable a paperless contract but are decentralised. They are used to manage transaction performance between parties. Block chain makes it secure and is tamper-proof. We can use and integrate smart contracts in Intellectual Property Rights (IPRs) as well. Intangible assets can be digitized and secured along with trust, transparency, and safety of both the enforcer and the consumer. In this paper we are going to see the Ethereum platform to develop custom based smart contracts.

Key words: Block Chain, Smart contracts, Lifecycle, Platform, Ethereum.

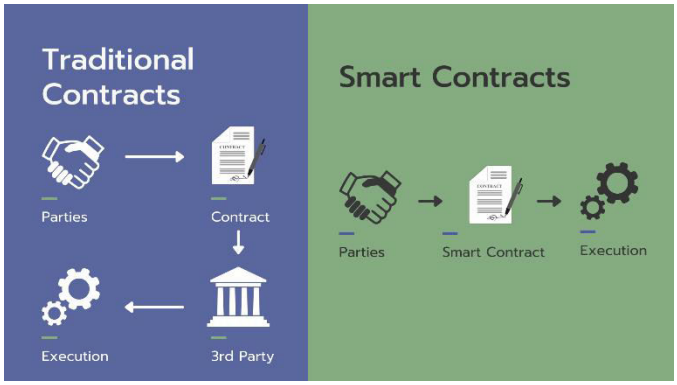
Introduction: Blockchain technology collects and holds information together in groups, known as blocks. The Bitcoin whitepaper (1) was published in the year 2008, Since then, blockchain technology has been used in different applications. Decentralization, trust, immutability, transparency are some of the important features of Block chain.

Smart contracts are one of the most relevant applications of blockchain .These contracts self-executing contracts containing the terms of the agreement between the parts. Ttrusted transactions and agreements are carried out among different anonymous parties. Ethereum smart contracts upload and execute code that carries out business logic to the blockchain [2, 3]. These code resides on a blockchain as multiple functions with unique address and can be called.

Smart Contracts: It is described as "an agreement whose execution is automated". A Smart Contract (or crypto contract) is a program that controls the transfer of digital assets between the parties under certain conditions automatically. A smart contract works in the same way as a traditional contract while also automatically enforcing the contract with the security coding of the blockchain. Thy are executed exactly as they are coded. Traditional contract is enforced by law, whereas smart contracts are enforced by code.

ISSN PRINT 2319 1775 Online 2320 7876

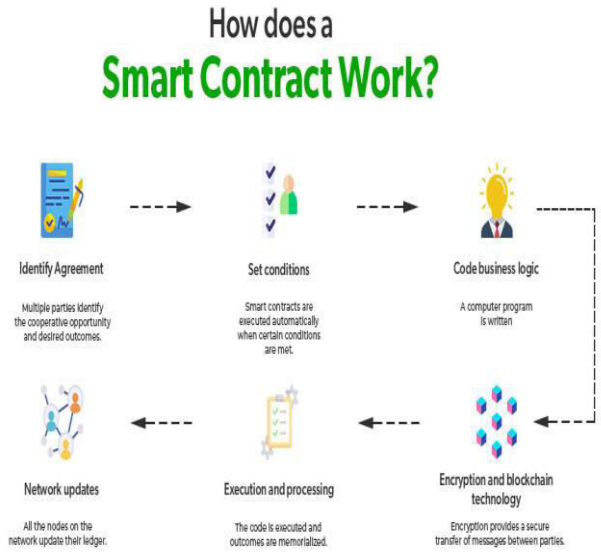
Research paper UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Oct 2022



Third parties are not involved in the performance of credible transactions. Even though we can track these transactions it is irreversible. It helps to provide security that is superior to the traditional contract law and in the meanwhile reduce other transaction costs that are associated with contracting.

Working:

- The details and permissions written in code with an exact sequence of events.
- The time constraints can be introduced to set deadlines in the contract.
- Each contract has its address in the blockchain. This is used as an identification of the contract has been broadcasted on the network.
- The logic used is IF-THEN .



First Agreement is reached among different parties. Smart contracts could be activated by parties themselves or when certain conditions. When the condition is met the computer program which is written will be executed automatically. Information about the contacts are sent securely with proper authentication. The outputs are stored for compliance and verification. After smart contracts execution all the nodes on the network update their ledger to reflect the new state.

Life Cycle:

Creation: First negotiation is done on the obligations, rights and prohibitions on contracts. Parties after drafting an initial contractual agreement. This agreement written is converted into a smart contract written in computer languages including declarative languages and logic-based rule languages.[5]

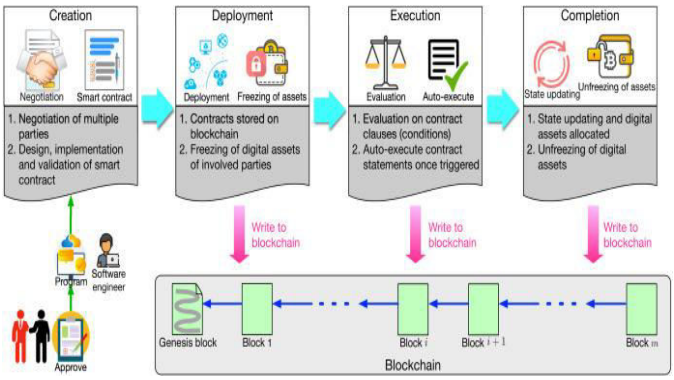
ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Oct 2022

Deployment: Once the smart contracts are validated, they are deployed to platforms on top of blockchains. Contracts stored on the blockchains cannot be modified due to the immutability of block-chains. [5]

Execution: After the deployment of smart contracts, the contractual clauses have been monitored and evaluated. Once the contractual conditions are reached, the contractual procedures (or functions) will be automatically executed.[5]

Completion:. After a smart contract has been executed, new states of all involved parties are updated. During the execution of the smart contracts the updated states are stored in blockchains.



Smart Contract Platform

Smart contracts are developed on platforms. They provide interface to develop applications. We have five main platforms: Ethereum, Hyper ledger fabric, Corda, Stellar and Root stock.

Ethereum is a decentralized computing platform. It has developed *Turing-complete* languages such as Solidity, Serpent, Low-level Lisp-like Language (LLL) and Mutan.In thi platform each participant is identified by digital wallet. It uses EVM to run contracts.[6]

Hyperledger Fabric is a distributed ledger platform. It uses Docker container to run instead of EVM.It supports high-level programming languages such as Java and Go (*aka* Golang).[7]

Corda is mainly used for digital-currency applications. It uses distributed-ledger platform for saving and processing historical digital-asset records. It uses high-level programming languages such as Java and Kotlin.[8]

Stellar It is similar to Corda, it is a specialized platform for digital-currency applications. It is simpler and more accessible. Stellar supports languages such as Python, JavaScript, Golang and PHP. [9]

Rootstock runs on top of Bitcoin, with faster execution of transactions. The data model of is account-based .It is compatible with Ethereum.[10]

We are going to discuss in this paper on implementation of smart contract using Ethereum.

Ethereum

In 2013, Vitalik Buterin proposed Ethereum. It as the ability to build and run decentralized applications with smart contracts. The core of Ethereum is the Ethereum Virtual Machine (EVM) [13]

Application crated using existing programming languages that can run on Ethereum virtual machines, such as Javascript, Python, etc. To maintain the consistency of the entire blockchain,

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Oct 2022

each network node runs an Ethereum virtual machine. It is fault tolerant, assures zero downtime, and data stored on the blockchain are immutable and anti-censorship.

Three important aspects of Ethereum smart Contracts

1. Execution Context
2. Gas
3. Immutability

Execution Context

Since Smart Contracts run in an isolated fashion they can only see data available on the Blockchain or call other Smart Contracts; they cannot make calls to any service or query data from the outside.

Gas

Running the code in EVM is chargeable which is called as Gas. Which is short fractions of Ether .Gas is paid for every transaction we do.

Immutability

Their definition (byte code) cannot be changed or updated once they are deployed.If we need to change , we have to deploy a new version of smart contracts in a new address.

Ethereum Virtual Machine (EVM)

EVM is a special virtual machine that's used as an interpreter for the assembly language. The EVM's functionality is very limited .

Solidity

Solidity is a general-purpose programming language developed on top of the EVM. It uses a class (*contract*) and methods that define it. It allow you to perform arbitrary computations, but its main purpose is to send and receive digital tokens as well as store states.

Design Smart contracts are compiled in solidity [14] and deployed on the Geth Ethereum client. Dapp is used to connect and submit transactions when Ethereum node connected to the Blockchain exposes an RPC-JSON interface over HTTPS or Web Sockets.

When the contract is created, we use some variables and two structures.

1.DO is the creator of the contract, and its Ethereum account address is recorded as owner. All Ethereum account addresses that call smart contracts are recorded as msg.sender .

2.The file ID hash is used as an index of file information, and is recorded as HashFileId recorded as userAddress.

3. Two structures: file information named File and user information is stored in User. HashFileId is used to store file related information and userAddress stores User related information.

IN addition to this we have 8 interfaces named Storage SC.

The following is the sample code of smart contracts:

```
pragma solidity >=0.8.4;

contract MyToken {
    mapping (address => uint) balances;
    event Transfer(address indexed _from,
address indexed _to, uint256 _value);
```

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Oct 2022

```
    constructor()  {
        balances[tx.origin] = 10000;
    }
    function sendToken(address receiver,
uint amount) public returns(bool success) {
        if (balances[msg.sender] < amount)
return false;

        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Transfer(msg.sender, receiver,
amount);
        return true;    }

    function getBalance(address addr) public
view returns(uint) {
        return balances[addr];
    }
}
```

MyToken is the name for the contract, It is used to reference this contract in code by other contracts. mapping is a construct that acts as a Dictionary or Hash for key/value pairs. Transfer is an event emitted by the contract whose payload contains two addresses (from/to) and a value..sendToken moves tokens from one address (the owner) to another address (treceiver address).getBalance() returns the balance in tokens associated with particular address.

Once the code is ready, we can start compiling the code. The solc compiler generate a binary representation of the contract for deployment in the Blockchain and a programmatic interface.

The output from the solc compiler is deployed in any of the available Ethereum main networks like mainnet and testnet or in developer emulators that behave like regular nodes and expose the same JSON-RPC interface. In order to deploy in real Ether we need to pay for Gas.

Once the contract is successfully deployed in the emulator we can call the methods. We call getBalance(), sendToken() by using the contract instance .

Steps to Deploy Ethereum Smart Contracts

1. We need to switch to the main Ethereum network at MetaMask to go live.
2. Add some real Ethers.
3. Deploy your smart contract using remix
4. After successfully deployed, we can search smart contract in this http://www.etherscan.io and select the one we need.
5. Verify your smart contract here, click **verify contract**.
6. Pate the code in Etherscan.
7. Check **optimization** to **Yes** or No based on selection in remix; otherwise, select **No**.
8. Click **Verify**.
9. It will take a few minutes, and your smart contract will be live if no issues occur.

Once the contract is live the status is updated in the ledger and its available for the concerned parties.

Conclusion

The Ethereum blockchain's smart contracts is powerful tool for automating but they have limitations. Some of the limitations are their complexity, immutability, cost, scalability challenges, and integration with legacy systems. Despite these limitations, the Ethereum blockchain are expanding, and the benefits of using smart contracts can outweigh their drawbacks.

ISSN PRINT 2319 1775 Online 2320 7876

Research paper UGC CARE Listed (Group -I) Journal Volume 11, Iss 10, Oct 2022

References

1. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. White paper Google Scholar
2. Toyoda, K., et al.: A novel blockchain-based Product Ownership Management System (POMS) for anti-counterfeits in the post supply chain. IEEE Access PP(99), 1 (2017) Google Scholar
3. 'Ethereum'. http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html/. Accessed 20 Dec 2017
4. De Filippi P and Wright A, Blockchain and the Law: The Rule of Code (Harvard University Press 2018), p.74.
5. Sillaber C., Waltl B. Life cycle of smart contracts in blockchain ecosystems Datenschutz Datensicherheit - DuD, 41 (8) (2017), pp. 497-500
6. Buterin V., *et al.* Ethereum White Paper Ethereum (2013) URL https://www.ethereum.org/
7. C. Cachin, Architecture of the hyperledger blockchain fabric, in: Workshop on Distributed Cryptocurrencies and Consensus Ledgers, 2016.
8. Brown R.G.The corda platform: An introduction https://www.corda.net/content/corda-platform-whitepaper.pdf
9. Mazieres D. The Stellar Consensus Protocol: A Federated Model for Internet-Level Consensus
10. Stellar Development Foundation (2016)
11. URL https://www.stellar.org/papers/stellar-consensus-protocol.pdf
12. Lerner S.D. Rootstock whitepaper URL https://docs.rsk.co/RSK_White_Paper-Overview.pdf
13. W.Gavin,"Ethereum: A secure decentralised generalised transaction ledger", 2014, [online] Available: https://ethereum.github.io/yellowpaper/paper.pdf.
14. C.Dannen, Introducing Ethereum and Solidity: Foundations of Cryptocurrency and Blockchain Programming for Beginners, New York, NY, USA:Apress, 2017.
15. https://www.geeksforgeeks.org/smart-contracts-in-blockchain/
16. Steps to Create, Test, and Deploy an Ethereum Smart Contract - DZone
17. 101 Smart Contracts and Decentralized Apps in Ethereum (auth0.com)